TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Kevin Nemeržitski

Sünkroniseeritud dokumentide kuvaja arendus nutiseadmetele

Bakalaureusetöö (9 EAP)

Juhendajad: Kaarel Kruus
Jüri Kiho

Tartu 2016

Sünkroniseeritud dokumentide kuvaja arendus nutiseadmetele

Lühikokkuvõte:

Muusikaõpetaja informeerib õpilast olulisest kohast noodil enamasti suuliselt. Selle asemel võiks kahel nutiseadmel kuvada sama nooti digidokumendina ning dokumendi peale märkme tegemisega saaks viidata olulisele kohale. Kontsertidel on tihtipeale solisti kõrval abiks assistent, kes keerab noodi lehekülgi. Paljude jaoks on see kontserdil häirivaks teguriks. Mobiilset tehnoloogiat kasutades saaks assistent diginoodi lehekülgi keerata eemal (näiteks publikus) olles. Käesoleva mobiilirakenduse arendamine tuleneb vajadusest hõlbustada eelnimetatud tegevusi. On juba olemas mobiilirakendusi, mida saaks kasutada, aga neil on teatud puudusi. Töös on loodud multiplatvormiline lahendus, mis sünkroniseerib dokumenti ja selle märkmeid sellel. Seadmeid saab ühendada, kasutades erinevaid tehnoloogiaid (Bluetooth, WiFi), ja ühenduse katkemisel ühendus taastatakse. Rakenduse taasavamisel taastub eelnev seis. Võõras osapool ei saa ühendust mõjutada ega vahele segada, kuna ühendus on krüpteeritud. Seadmed suudavad üksteist iseseisvalt autentida.

Võtmesõnad:

Sünkroniseeritud dokument, mitmetehnoloogiline võrguühendus, ühenduse krüpteerimine, iseseisev autentimine, mobiilirakendus

CERCS: T120

Synchronized document viewer development for mobile devices

Abstract:

Music teacher informs a student about a music sheet mostly verbally. Instead they could use two mobile devices to display the digital sheet and make remarks on the document. During concerts, musician may have an assistant for turning sheet pages. Many consider this disruptive. Instead, using mobile technology, assistant could use a digital sheet to turn pages within distance (e.g. within audience). This thesis mobile application development aims to resolve issues mentioned above. Existing applications could be used, but those have shortcomings. A multiplatform solution was developed, which synchronizes a document and remarks on it. Devices can connect using multiple technologies (Wi-Fi, Bluetooth). Unexpected disconnects are restored automatically. The application is able to restore its synchronization state if restarted. Third party isn't able to modify or interrupt the connection, because the connection is encrypted. Devices are able to authenticate each other without a centralized unit.

Keywords:

Synchronized document, multiple technology network connection, connection encryption, independent authentication, mobile application

CERCS: T120

Sisukord

1. Sissejuhatus

1.1 Probleemi kirjeldus

Sihtvaldkond

Kõige üldisemas plaanis on käesoleva töö sihtvaldkonnaks infotehnoloogiliste vahendite rakendamine inimeste kollektiivse tegevuse toetamiseks. Konkreetsemalt peame silmas eeskätt muusika esitamise või harjutamise protsessi. Sel puhul on kollektiivse tegevuse osapoolteks orkestrandid, lauljad, dirigendid kui ka muusikaõpetajad, õppijad jt kuni assistentide-noodilehekeerajateni välja. Spetsiifiliste joontena võib siin välja tuua järgmised momendid.

1. Osalejate ruumiline lähedus: reeglina vähem kui 10 m (erandiks nt ühendkoorid).
2. Ajakriitilisus: muusikapala esitamise käigus peavad osalejatevahelised kommunikatsiooniaktid toimima reaalajas, sekundi murdosa jooksul.

Sama sihtvaldkonda on peetud silmas ka Kaarel Kruusi magistritöös [1]. Valdkonna illustreeriva tutvustamise huvides esitame alljärgnevas mõned tsitaadid K. Kruusi tööst:

- „[…] musical pieces usually contain repetitions, which also reflects in the score physical layout and this can sometimes lead to intense page turnings. “ ([1], lk 7).
- „[…] human assistance (standing or sitting next to the artist) is used for the page turning task” ([1], lk 8).
- „[…] reaalajas muusika-alast kaastööd võimaldavate vahendite hulk praktiliselt olematu“ ([1], lk 3).

Tarkvarasüsteem Partiture

Tegemist on K. Kruusi poolt välja töötatud süsteemiga [1], mis lahendab mõned esmased probleemid muusika esitamise või harjutamise protsessi tehnoloogilise toetamise osas. Loodud tarkvarasüsteemi jaoks on läbi viidud nõuete analüüs, arhitektuuri kirjeldus, mobiilirakenduse arendus ja selle valideerimine. Süsteem koosneb kahest osast:

- veebirakendus – kasutajate ja nootide haldamise süsteem
- mobiilirakendus – nootidega töötamiseks.

Kui solistil ja assistendil mõlemal on olemas nutiseade, kus kuvatakse sama nooti digitaalselt, mis on sünkroniseeritud üle juhtmevaba võrgu, siis saaks assistent digikujul oleva noodi lehekülgi keerata ilma, et ta oleks solisti kõrval.

Loodud mobiilirakendus Partiture on Apple iOS (edaspidi iOS) platvormi spetsiifiline ja kasutab ainult iOS-il toetatud teeke.

Probleemi täpsustus

Käesolev töö kirjeldab multiplatvormilise rakenduse arendamist nutiseadmetele, mis on baasiks Partiture mobiilirakenduse edasiarendusele. Rakenduse loomisel on põhirõhk PDF failide kuvamisel/muutmisel, WiFi ja Bluetooth krüpteeritud stabiilsel ühendusel seadmete vahel, lihtsal dokumentide haldamisel ning dokumendi sünkroniseerimisel kahe seadme vahel. Rakendus on kasutatav kaheliikmeliste kollektiivide korral, nagu õpetaja + õpilane, interpreet + lehekeeraja, (klaveri)duo jmt.

1.2 Eesmärk

Eesmärk on luua multiplatvormiline mobiilirakendus, mis töötaks Google Androidil (edaspidi Android) alates versioonist 4.0 ja iOS-il alates versioonist 7.0.

Rakendus peab kuvama dokumenti, mille visuaalset esitust sünkroniseeritakse teise seadmega. Sünkroniseerima peab sama dokumendi näitamist, lehekülje pööret ning kasutaja poolt dokumendi peale tehtud märkmeid.

Omavahel sünkroniseeritud kasutajaid (rakendusi) on piiratud kaheni. Ühendus peab olema mõlemasuunaline. Mõlemad kasutajad saavad dokumenti mõjutada.

Toetatud ühendused on WiFi ja Bluetooth, mis peavad aitama üksteisel ühendust luua. Juhtmevaba ühendus on teatavasti ebastabiilne; katkemisel tuleb see automaatselt taastada ja dokument sünkroniseerida.

Kasutada tuleb väiksema latentsiga ühendust andmesidel. Dokumentide sünkroniseerimise ajal peab edastatav andmemaht olema minimaalne.

Ühendused seadmete vahel tuleb krüpteerida ning veenduda, et ollakse tegelikult ühenduses seadmega, millega ühenduda taheti. Võõras seade ei tohi saada ühendust segada. Kasutajal peab olema kontroll selle üle, mis seadmega ühendutakse.

1.3 Skoop

Bakalaureusetöö piiratud mahu tõttu viiakse rakenduse arendus lõpuni vaid Android platvormile. Arendusel peeti silmas ka iOS platvormi; iOS spetsiifilise arenduse juhised on välja toodud 4. peatükis.

1.4 Töö struktuur

Töö koosneb kümnest peatükist ja lisadest. Käesolev esimene peatükk annab nii ülevaate probleemist kui ka ülesande püstitusest. Teises peatükis on välja toodud loodavale rakendusele sarnased realisatsioonid ning käesoleva töö raames loodava rakenduse eelised. Kolmandas peatükis kirjeldatakse rakenduse arendusvahendeid ning põhjendatakse nende valikut. Neljandas peatükis tuuakse välja rakendusele vajalikud muutused, et see töötaks iOS platvormil. Viiendas peatükis selgub, kuidas kuvada PDF faile seadme ekraanil. Kuuendas peatükis kirjeldatakse, kuidas seadmete vahel luuakse sünkroniseeritud ühendus. Seitsmendas peatükis on kirjas dokumendi sünkroniseerimiseks vajalikud elemendid ning kuidas neid teha. Kaheksandas peatükis analüüsitakse ühenduse kiirust ja stabiilsust eraldi WiFi ja Bluetoothi jaoks ning mõlemat korraga. Üheksandas peatükis kirjeldatakse, kuidas tagati töö kvaliteet. Kümnes peatükk on töö kokkuvõte. Lisadesse on paigutatud: arendatud tarkvara ning selles imporditud osade litsentsid ja teadaolevad vead, kasutusjuhend, Androidile kompileeritud rakendus ja selle installeerimisjuhend.

2. Sarnased rakendused

2.1 RemoDroid

RemoDroidiga [2] saab kaugjuhtida Androidi seadet teise Androidi seadmega või personaalarvutiga. Ekraani jagamisel teisele seadmele saadetakse kaadripilt pideva andmevoona. Teine seade kuvab saadud kaadripildi enda ekraanile ning saab juhtida esimest seadet.

Ekraani jagava Androidi seadmel peab olema rakendusele antud juurõigused.

Toetatud ühendused on WiFi ja 3G. Ekraani jagamisel võib tekkida probleeme, kui ekraanipilti jagava seadme ekraan on suurem ühenduvast seadmest. Ühenduse katkemisel seda ei taastata.

2.2 AirDroid

AirDroidga [3] on võimalik lugeda ja saata kõikvõimalikke faile. Seadmega ühendumine toimub läbi Chrome brauseri. Androidi seadme ekraanipildi jagamiseks teisele seadmele saab kasutada *Screenshot* akent, mis saadab aeg-ajalt pilte kaadrist. Ekraanipilti jagavat seadet kaugjuhtida läbi brauseri Androidil ei saa. Personaalarvutite jaoks on olemas AirDroid Desktop, millel on olemas AirMirror, millega saab ka teist seadet kaugjuhtida. AirDroid Desktopi kasutamise jaoks peab kasutaja looma.

Ekraani jagava Androidi seadmel peab olema rakendusele antud juurõigused.

Kuna ühendus käib läbi brauseri, siis saavad ühenduda seadmed, mis on Internetti ühendatud. Ühenduse katkemisel seda ei taastata.

2.3 TabletRemote

TabletRemote [4] toetab kaugjuhtimist läbi sisseehitatud sisendite. Selleks on näiteks nupud nagu „Back“„ ja „Home“. Teisel seadmel saab navigeerida DPadiga. Vabalt sõrmega tehtavaid sisendeid rakendus ei toeta. On mõeldud rakenduste kontrollimiseks, millel on kindlat tüüpi sisendid.

Toetatud ühendused on lokaalne WiFi ja Bluetooth.

Ühenduse katkemisel seda ei taastata.

2.4 Teamviewer Host ja Teamviewer for Remote Control

Teamviewer [5,6] toetab teise seadme kaugjuhtimist. Kahe Android seadme vahel kaugjuhtimist läbi viia ei saa. Töötab vaid personaalarvuti ja Android seadme vahel. Üle ühenduse saadetakse seadme ekraanipildi kaadreid.

Ühendus seadme ja arvuti vahel toimub läbi Interneti.

2.5 Loodava rakenduse eelised

Kõik sarnased olemasolevad rakendused on mõeldud kogu seadme sünkroniseerimiseks, mistõttu edastatakse ühenduvale seadmele kogu ekraanipildi kaader. See suurendab latentsi sünkroniseerimisel. Loodav rakendus on spetsialiseeritud dokumendi sünkroniseerimisele. Seega ei edastata ekraanipilti, vaid mõlemas seadmes on olemas kuvatav dokument ning sünkroniseeritakse vaid lehekülje pöördeid ja märkmeid. Latents sünkroniseerimisel on väiksem. Kuna töödeldakse vähem andmeid, kasutatakse ka vähem seadme ja ka võrgu ressursse.

Enamik sarnastest realisatsioonidest toetavad vaid WiFi ühendust, kusjuures ühendus on personaalarvuti ning nutiseadme vahel. Käesolev rakendus toetab nii WiFit kui ka Bluetoothi kahe nutiseadme vahel ning on võimeline iseseisvalt taastuma.

Käesolev rakendus ei vaja Androidi süsteemi juurõigusi. Kõik olemasolevad rakendused, välja arvatud Teamviewer, vajavad ligipääsu juurõigustele.

3. Arendusvahendid

Arendus toimub Windows 8.1 64-bitises operatsioonisüstemis. Rakendust luuakse Qt [7] raamistikuga 5.5, millega on autor eelnevalt tutvunud. Kuna eesmärk on luua rakendus nii Androidile kui ka iOSile, siis selleks sobib hästi Qt, mis on mõeldud multiplatvormiliseks arenduseks. Teine potentsiaalne raamistik on Xamarin [8] Visual Studiole.

Qt raamistiku tõttu on programmeerimiskeeleks C++, ka programmi loogika väljendub C++ keeles. Kasutajaliides on realiseeritud QML keeles.

Versioonihaldustarkvara ei kasutata, kuna arendust viib läbi üksnes autor. Loomulikult tehakse koopiaid projekti kaustast olulistel hetkedel.

3.1 Android

Kasutatakse järgnevaid Qt seadetes määratavaid vahendeid: Java JDK 1.8.0_45, Android NDK r10e ja Apache Ant 1.9.6.

Arendamisel kasutatakse testimiseks nutiseadmeid Samsung I9301I Galaxy S3 ja LG Optimus L3 II E430.

3.2 Teekide kompileerimine

OpenSSL 1.0.2g lähtekoodi kompileerimiseks kasutatakse MinGW msys 1.0 (GCC versiooniga 4.9.3) [9].

MuPDF 1.3 [10] lähtekoodi kompileeritakse Windowsi käsurealt.

4. Rakendus iOSil

Rakendus iOSile ei ole täielikult valmis. On teatud puudused, mida saab lahendada vaid Mac OS X operatsioonisüsteemil töötavate arendusvahenditega. Järgnevad mõned juhised üleminekuks iOSile.

Tuleb lisada iOSis olemasolev PDFi konverteerija. Selleks tuleb implementeerida klassi *IOSPdfRenderer* funktsioonid. Koodi kirjutamine spetsiaalselt iOSile on programmeerimiskeeles Objective-C. C++ ja Objective-C on otse Qt-s toetatud. Päisefailid on mõlemal samad. Objetive-C jaoks on *cpp* laiendusega failide asemel laiendusega *mm* failid.

Teek OpenSSL tuleb kompileerida eraldi iOS platvormile. Kompileeritud OpenSSL peab olema lisatud teegina projekti *pro* faili.

5. PDFi kuvamine

Qt ei toeta PDFi kuvamist sisse ehitatud teekidega nii, et seda saaks muuta. On olemas eraldi brauseri akna läbi PDFi kuvamine, aga selle peale ei saa teha märkmeid. Märkmete tegemiseks peab konverteerima PDFi pildiks. Kuna see protsess on mahukas, siis kasutatakse olemasolevaid vahendeid või teeke.

Lõpptulemusena kuvab rakendus PDFi pildina kasutades QMLi *Image* klassi. Pildi andmete edastamine QMLi käib läbi *QQuickImageProvider* klassi, mida realiseerib *ImagesViewImageProvier* klass. Lehekülje konverteerimine on aeganõudev. Seepärast on prioriteet konverteerida enne dokumendi vaates avatud leheküljele lähedasemad leheküljed.

5.1 iOS

iOS versioonis 7 on PDF konverteerimine sisse ehitatud. Rakenduses pole see realiseeritud.

5.2 Android

Android toetab PDFi konverteerimist alates versioonist 5.0. Seega on vaja kasutada kolmanda osapoole teeki, et saaks PDFi kuvada ka alates versioonist 4.0. Selleks sobib teek MuPDF, mis annab väljundiks pildi RGBA baidijada. Rakendus kasutab kuni 5.0-ni MuPDFi, edaspidi aga sisseehitatud konverteerijat.

Androidi sisseehitatud PDF konverteerija kasutamiseks tuleb C++ koodis välja kutsuda Java klasse ja meetodeid. Selle jaoks on Qt-s olemas *QAndroidJniObject* klassi staatilised funktsioonid. Rakenduses on vastavad väljakutsed realiseeritud *AndroidPdfRenderer* klassis, mis suhtleb otse Java koodiga. Kui Android seadme versioon on <5.0, siis kasutatakse MuPDFi. Kuna MuPDF on kompileeritud keeles C, siis kutsutakse Javas selle funktsioone Java *native* kutsetega, mis on *MuPDFCore* Java klassis. Alates versioonist 5.0 kasutatakse Java koodis sisseehitatud PDF konverteerijat, mis on realiseeritud *BuiltInPdfRenderer* Java klassis.

6. Ühendus

Seadmeid tuleb üksteisest eristada. Selle jaoks määratakse igale seadmele identifikaator (edaspidi SID). SID on rakenduse alustamisel genereeritud sertifikaadi SHA1 räsi. Igal seadmel on olemas ka ekraanile kuvamiseks mõeldud nimi, milleks on Bluetooth seadme nimi.

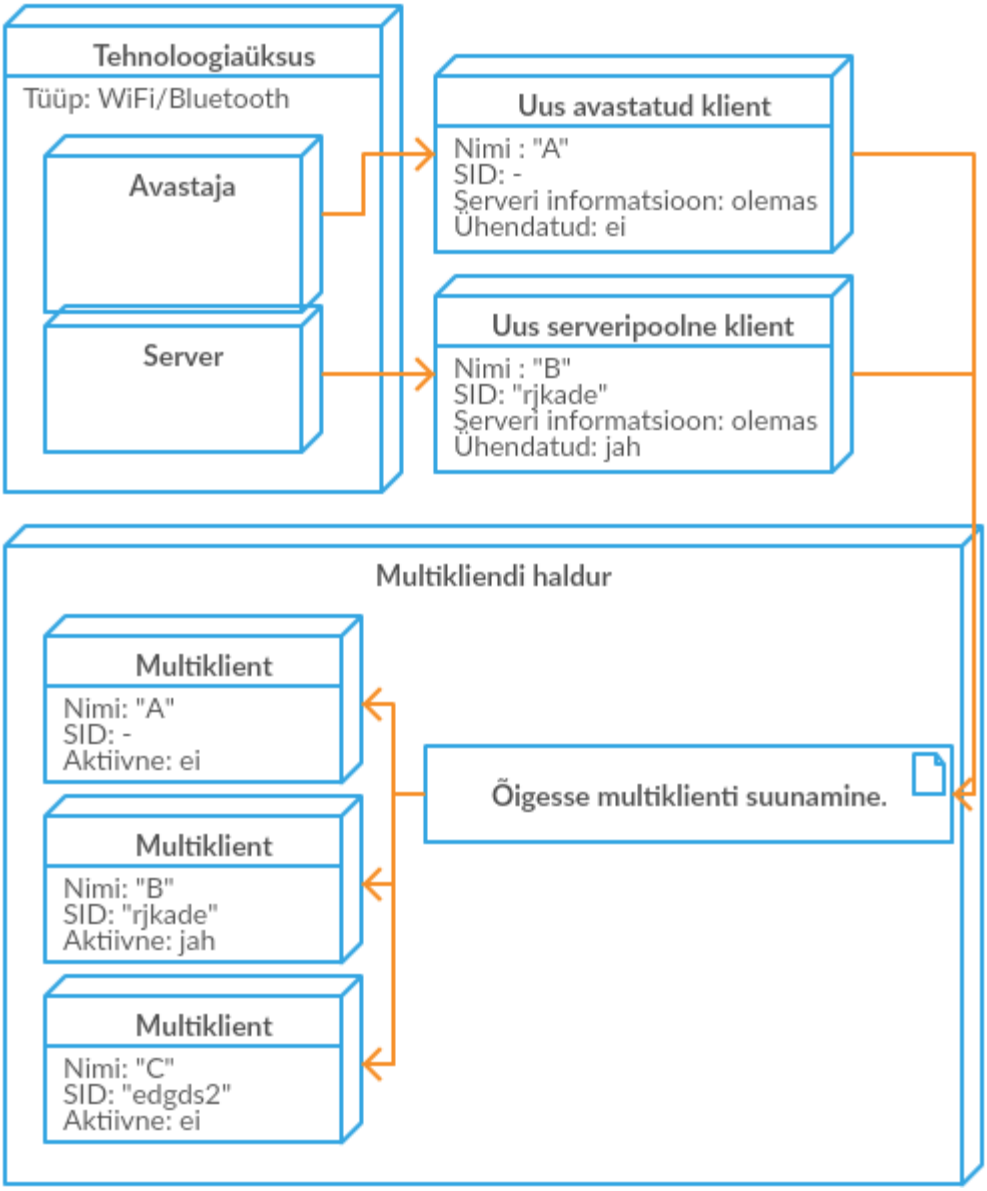
Ühenduse loomisel on eristatavad neli objekti: server, klient, serveripoolne klient ja avastaja. Avastaja ülesanne on otsida lähedasi seadmeid, kus rakenduses server töötab. Kui avastaja saab teada, kuidas serveriga ühenduda, siis kuvatakse andmed ekraanil. Kasutaja alustab serveriga ühenduse loomist kasutades klienti. Kui serveriga ühenduse loomine õnnestub, siis loob server serveripoolse kliendi. Andmeside vahetusel on seega kaks osapoolt: klient ja serveripoolne klient. Rakendusega ühenduses olevat teist poolt nimetatakse paariliseks (*peer*).

Komplekti, mis koosneb avastajast, serverist ja kliendist, mis kasutavad sama tehnoloogiat (WiFi, Bluetooth), nimetatakse tehnoloogiaüksuseks.

6.1 Ühenduste haldamine

Kõik kliendid, mis ühenduvad serveriga, tuleb vastu võtta ning ära töödelda – ühtegi klienti ei tohi ignoreerida. Kõikide klientide haldamiseks on multikliendi haldur, mis hoiab iga seadme ühendusi eraldi multikliendina, vt joonist 1. Serveripoolne klient suunatakse halduri poolt vastavasse multiklienti (multiklient on üksüheses vastavuses SID-ga), mis otsustab, kas klient vastu võtta või mitte.

Aktiivne multiklient on see, millele kasutaja on andnud nõusoleku ühenduse loomiseks. See tähendab, et kui aktiivne multiklient on olemas ning saadud serveripoolne klient suunatakse mõnesse muusse multiklienti, siis ühendus peatatakse. See väljendab rakenduse omadust lubada ühendumist vaid ühe seadmega.



Joonis[1] 1. Uute klientide haldamine

Joonisel 1 on näiteolukord, kus avastati klient A ja serveriga ühendus klient B. Klient A suunatakse multiklienti A ning ekraanile ilmub uus seade nimega A. Klient B suunatakse multiklienti B ning koheselt jätkatakse ühendust, sest multiklient B on aktiivne (kasutaja andis nõusoleku ühenduse loomiseks).

6.2 Serveri avastamine

Bluetooth

Bluetooth korral kasutatakse teenuse avastamise protsessi (SDP). Kui luuakse Bluetooth server, siis registreeritakse sellega seotud teenus SDP-sse. SDP hoolitseb selle eest, et teised seadmed suudaksid teenust avastada. Qt-s on olemas klass *QBluetoothServiceDiscoveryAgent*, millega saab SDP-sse registreeritud teenuseid leida.

[1] Kõik joonised käesolevas peatükis on valmistatud veebitarkvaraga Creately; https://creately.com/ (11.05.2016).

Teenuse leidmiseks peab olema seade avastatavas olekus. Android saab olla avastatavas olekus ajutiselt, ainult siis, kui kasutaja selleks nõusoleku annab. Avastatav olek kestab kuni 300 sekundit. Aja lõppedes muutub olek ühendatavaks ning enam seadet avastada ei saa.

iOS seadmed on alati avastatavas olekus (kui Bluetooth on sisse lülitatud).

QBluetoothServiceDiscoveryAgent annab otsimise tulemusena teenuse informatsiooni, mis alati sisaldab Bluetooth nime ja Bluetooth MAC aadressi. Kuna SID-d ei teata, siis ajutiselt toimub identifitseerimine seadme nime järgi.

Teenuse avastamine toimib ka siis, kui seadmed pole Bluetooth paari moodustanud, paari loomise nõue tekib, kui server ja klient ühenduse loovad.

WiFi

Serverite avastamiseks kasutatakse UDP protokolli ja võrgu *broadcasti*. Programm vahetab andmeid datagrammidena. Kuna kuulatakse kõiki pakette teatud pordis, siis on vaja filtreerida võõraid pakette. Iga datagrammi algusesesse lisatakse 64 bitine number, mis on kõikidel seadmetel ühine.

Kui seade A soovib avastada servereid, siis saadetakse *broadcast* datagrammi, mis tähendab järgnevat: „otsin servereid, vasta mulle, kui sul on server olemas“. Kõik seadmed, millel server töötab ja on avastatavas olekus, saadavad vastuse, mis sisaldab seadme Bluetooth nime, WiFi IP aadressi ja serveri porti. Ajutiselt identifitseeritakse seadet nime järgi.

6.3 Ühenduse loomine

Teise seadmega ühenduse loomiseks tuleb seade avastada vähemalt ühe korra, et tuvastada seadet nime järgi. Ühendudes otsustatakse, kas olemasolev serveri informatsioon on õige. Kui ei ole, siis alustatakse avastamist automaatselt enne ühenduse loomist.

Bluetooth

Bluetooth ühenduses kasutatakse RFCOMM protokolli. RFCOMM on ainus, mida Qt Androidil toetab. Ühenduse loomisel kasutatakse seadme Bluetooth MAC aadressi ja teenuse identifikaatorit. See informatsioon on püsiv ning tuleb avastada vaid üks kord.

WiFi

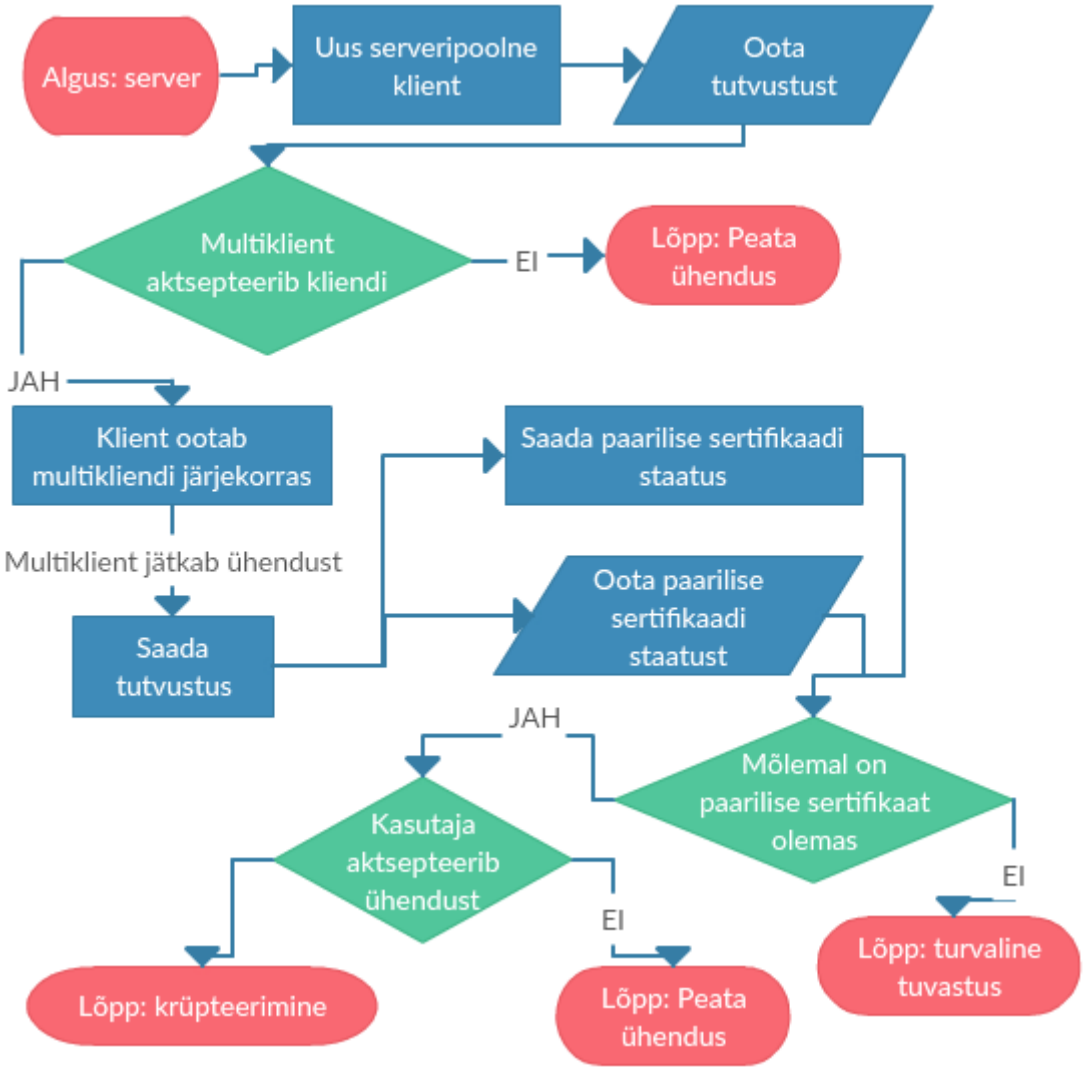
WiFi ühenduse protokoll on TCP. Saab kasutada ka UDP protokolli, aga vajalik ühendus peab olema usaldusväärne. Oluline on pakettide järjekord ja kohalejõudmine. Hiljem asendatakse TCP protokoll SSL protokolliga, mis on krüpteeritud TCP ühendus. Serveriga ühenduse loomiseks on vajalik seadmele määratud võrgu IP aadress ja serveri port. See informatsioon on muutuv ning tuleb avastada uuesti, kui WiFi võrk vahetub või server taaskäivitatakse.

6.4 Tutvustus

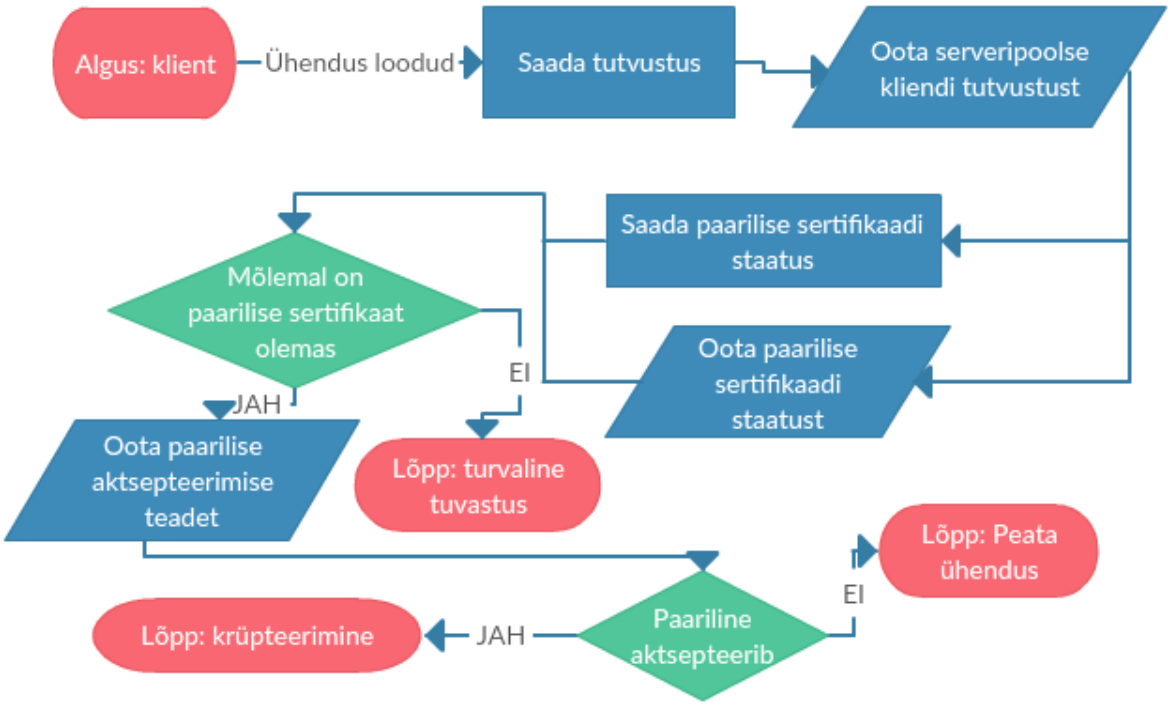
Ühendus on loodud. Server saab uue ühenduse ja luuakse serveripoolne klient, vt joonist 2. Kõigepealt ootab serveripoolne klient, et klient ennast tutvustaks, vt joonist 3. Tutvustus koosneb järgnevatest andmetest: seadme nimi, SID ning serveri informatsioon. Kui serveripoolne klient on tutvustuse kätte saanud, siis otsustatakse nime ja SID järgi millisesse multiklienti vastav serveripoolne klient panna.

Kui multiklient saab uue ühenduse, siis see pannakse järjekorda. Multiklient töötleb ühte ühendust korraga. Kui multiklient alustab serveripoolset kliendi töötlemist, siis serveripoolne klient jätkab ühendust. Serveripoolne klient saadab kliendile enda tuvustuse, millest klient järeldab, et ühendust võib jätkata.

Nüüd otsustatakse, kas on vaja üksteist turvaliselt tuvastada. Mõlemad pooled saadavad teate, kas neil on paarilise sertifikaat olemas. Kui vähemalt ühel pole paarilise sertifikaati, siis tuleb turvaliselt luua ajutine krüpteeritud ühendus sertifikaadi saatmiseks. Kui mõlemal on sertifikaat olemas, siis kuvatakse serveripoolse seadme ekraanil kinnitust ühenduse jätkamiseks.



Joonis 2. Serveripoolne tutvustus



Joonis 3. Kliendipoolne tutvustus

6.5 Turvaline tuvastus

Seadmed on üksteist turvaliselt tuvastanud, kui on suudetud sertifikaat paarilisele saata ilma, et kolmas osapool oleks seda välja vahetanud. Selle jaoks peab kasutaja kontrollima, et saadeti õige sertifikaat.

Kõige lihtsamini teostatavaks lahenduseks on kuvada enda ja paarilise sertifikaati ekraanil. Kasutajad võrdlevad seadmetel näidatavaid sertifikaate ning veenduvad, et avalikud võtmed on samad. Selline lahendus ei ole kasutajasõbralik, kuna on palju andmeid, mida tuleb visuaalselt kontrollida.

Rakendus kasutab RSA 2048-bitist võtmepaari. See tähendab, et PEM kodeeritud kujul on avalik võti 451 baiti. Kui arvestada vaid võtme sisulist osa, siis on 256 baiti. Kahe avaliku võtme korral on kokku 512 baiti, mida tuleks ekraanil kuvada.

Kasutajasõbralikkuse tagamiseks on mõistlik kuvada arvu. Selleks võib olla 6-kohaline arv (edaspidi PIN). PIN peaks katma võimalikult suurt kogust baite. Qt-s on suurim toetatud täisarv 64-bitine (8 baiti). PINi saamiseks tuleb 8 baidisest arvust võtta jääk.

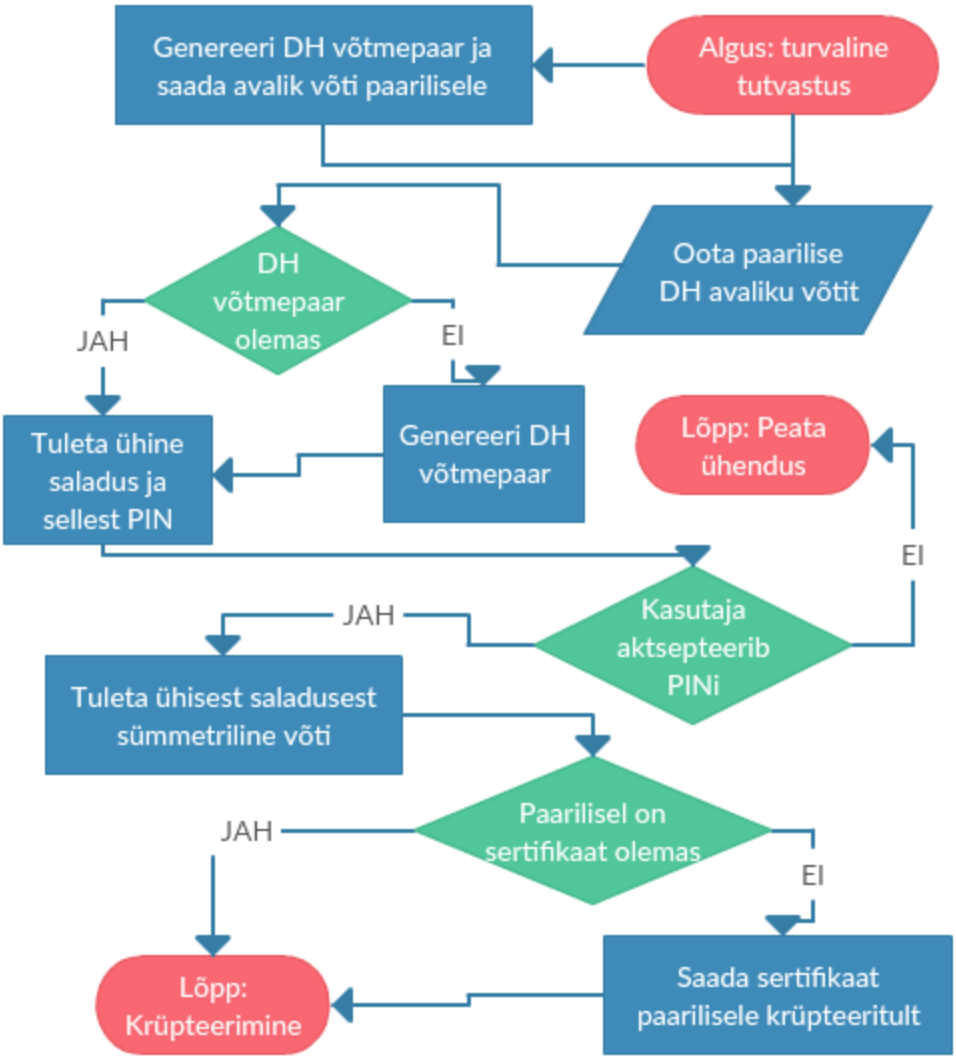
Üks võimalus 8 baidi saamiseks on võtta see avalikust võtmest. See tähendab, et kasutaja kontrolliga on kaetud 3% avalikust võtmest.

Teine võimalus on kasutada Diffie-Hellmani võtme vahetuse algoritmi (edaspidi DH). Mõlemad pooled genereerivad samade parameetritega võtmepaari, vt joonist 4. Avalik võti saadetakse paarilisele. Saadud avalikust võtmest ja enda salajasest võtmest tuletatakse ühine saladus, mis on paarilisel täpselt sama, kui saadi õige avalik võti kätte. Rakenduses loodav ühine saladus on 32 baiti. Ühisest saladusest võetakse 8 baiti, millest luuake PIN. Ühisest saladusest on kasutaja kontrolliga kaetud 25% baitidest.

Mõlemale kasutajale kuvatakse PIN ning mõlemad peavad veenduma, et mõlemas seadmes kuvatakse sama PIN. Enne tuvastamist ei jätkata kuni mõlemad pooled on PINi aktsepteerinud.

Nüüd saab ära kasutada ühist saladust sümmeetrilises krüptograafias, et saata paarilisele sertifikaat. Otse ühist saladust sümmeetrilise võtmena kasutada pole turvaline, sest see on jõumeetodil lahti murtav. See tähendab, et kui saata sertifikaat krüpteeritult ühise saladusega, siis kolmandal osapoolel piisavalt aega, et krüpteeritud sertifikaat lahti murda. Kui suudetakse see lahti murda, siis on teada, milline on ühine saladus, ning saab asendada saadetava sertifikaadi teisega. Jõumeetodi vastaseks kaitseks tuleb ühisest saladusest tuletada sümmeetriline võti, mille loomine peab olema aeganõudev. Sellisel juhul ei suudeta mõistliku aja jooksul jõumeetodil kõiki sümmeetrilisi võtmeid tuletada.

Mõlemal poolel on olemas sümmeetriline võti, et saata andmeid krüpteeritult. Eelnevalt jäeti meelde, kas paarilisel on sertifikaat olemas. Kui paarilisel sertifikaati ei ole, siis see saadetakse talle sümmeetrilise võtmega krüpteeritult.



Joonis 4. Turvaline tutvustus

6.6 Ühenduse krüpteerimine

Ühenduse krüpteerimiseks peab mõlemal poolel olema paarilise sertifikaat, mis saadakse turvalisel tuvastusel.

Bluetooth

Bluetooth ühendus krüpteeritakse juba Bluetooth paari loomise ajal. Eraldi pole vaja ühendust krüpteerida. Tuleb vaid kontrollida, et Bluetooth ühendus loodi seadmega, millega

ühendust taheti luua. Selleks tuleb veenduda, et paarilisel on olemas sertifikaadi avaliku võtmele vastav salajane võti.

Mõlemad pooled genereerivad suvalise tüki andmeid. Krüpteerivad selle paarilise avaliku võtmega ning saadavad paarilisele.

Saadud krüpteeritud andmetükk dekrüpteeritakse oma salajase võtmega ja krüpteeritakse paarilise avaliku võtmega ning saadetakse paarilisele tagasi.

Paariliselt saadud andmetükk dekrüpteeritakse oma salajase võtmega ning võrreldakse, kas saadud andmetükk on sama, mis esialgselt saadeti. Kui on sama, siis on kontroll edukas. Kui pole sama, siis ühendus katkestatakse.

WiFi

Qt-s on olemas *QSslSocket* klass, millega saab WiFi ühendust krüpteerida. See klass eeldab, et on olemas OpenSSL teek. *QSslSocketile* tuleb lisada enda salajane võti, enda sertifikaat ja paarilise sertifikaat. Kõigepealt alustab krüpteerimisprotsessi serveripoolne klient, aga enne selle alustamist saadetakse paarilisele teade, et serveripoolne krüpteerimine algas. Kui kliendipoolne klient saab teate kätte, siis see alustab kliendipoolset krüpteerimisprotsessi.

6.7 Aja sünkroniseerimine

Lehekülje pöörded toimuvad teatud hetkel. Kui paarilise saadakse lehekülje pöörde teavitus, siis on vaja otsustada, kas pööre vastu võtta või minema visata, olenevalt sellest, millal pööre läbi viidi. Kui enda seadmes on viimane pööre varasem, siis tuleb paarilise pöörde teavitus minema visata, vastasel juhul tuleb vastu võtta.

Enamasti ei ole seadmete ajatemplid samad, mis tähendab, et aega tuleb sünkroniseerida. Selleks piisab, kui leida seadmete ajatemplite erinevus, registreerides ajatemplid, ning tulemus jagada kahega [1].

Ajatemplite erinevuse leidmine

Leiame seadme A ja B vahelise ajatemplite erinevuse t_{A-B}. Tabelis 1 tähistab ühes reas seadme A aeg ja seadme B aeg sama hetke.

Tabel 1. Seadme A ja B vaheline andmeside t_{A-B} leidmisel.

Etapp	A aeg	B aeg	A sokkel	B sokkel
1	$t_A - p_o$	$t_B - p_0 - p_1$		Kirjuta: Alusta aja sünkroniseerimist
2	t_A	$t_B - p_1$	Loe: Aja sünkroniseerimise alustamise soov Kirjuta: Küsi B ajatemplit	
3	$t_A + p_1$	t_B		Loe: Küsitakse ajatemplit Kirjuta: Ajatempel t_B
4	$t_A + p_1 + p_2$	$t_B + p_2$	Loe: Ajatempel t_B Kirjuta: Ajatemplite vahe t_{A-B}	
5				Loe: Ajatemplite vahe t_{A-B}

Ideaaltingimustes on seadmete vaheline andmeside mõlemas suunas sama kiire, mis tähendab valemi $p_0 = p_1 = p_2$ kehtimist. Tegelikkuses on ühendus teatud ebastabiilsusega ning kehtib valem $p_0 \approx p_1 \approx p_2$. Kui andmesidet pole mõnda aega toimunud, siis tõenäoliselt kehtib valem $p_0 > p_1 \approx p_2$ (eelduseks on, et 1. kuni 4. etapp toimuvad viivitusteta). See tuleneb sellest, et võrguliides on vaja üles äratada, mis võtab aega. Selle tõttu on tabelis 1 lisatud 1. etapp, et ei tekiks olukorda, kus kindlasti kehtib võrratus $p_1 > p_2$.

Valemites kujul $y = [x_1 * x_2 * x_3 * \cdots * x_n]$, kus $*$ tähistab suvalist tehtemärki ja x_i tähistab arvu, on teada vaid y väärtus. x_i väärtus ei ole eraldi teada.

Vastavalt tabelis määratud A ja B aegadele saame ajatemplite erinevuse:

$$(1)\ t_{A-B} = t_A - [t_B - p_1].$$

Peale 4. etappi on seadmel A teada t_A, $[t_A + p_1 + p_2]$ ja t_B. Kuna $p_1 \approx p_2$, siis ainus viis valemis (1) $[t_B - p_1]$ väärtuse leidmiseks peame eeldama, et $p_1 = p_2$. Sellisel juhul $p_1 = \frac{p_1+p_2}{2}$.

$$(2)\ [t_B - p_1] = t_B - \frac{[t_A+p_1+p_2]-t_A}{2} = t_B - \frac{[p_1+p_2]}{2} = t_B - \frac{p_1+p_2}{2} = t_B - p_1$$

Valem (2) tähendab, et seadme B aeg 2. etapis on seadmelt B loetud ajatempel t_B miinus andmeside latents.

Kokkuvõtteks saame, et seade A saab leida ajatemplite erinevuse t_{A-B} valemiga (3).

$$(3)\ t_{A-B} = t_A - \left(t_B - \frac{[t_A+p_1+p_2]-t_A}{2}\right).$$

Peale t_{A-B} leidmist saadetakse see seadmele B (5. etapp), mis jätab selle meelde miinusmärgiga ehk $-t_{A-B}$.

Ajatemplite erinevus t_{A-B} on teatud veaga, mille väiksus sõltub sellest, kui täpselt kehtib võrdus $p_1 \approx p_2$. Mida suurem on p_1 ja p_2 erinevus, seda suurem on viga ajatemplite erinevuses t_{A-B}.

Enamasti kasutavad nutiseadmed Internetist võetud automaatselt uuenevat kella. See tähendab, et t_{A-B} väärtus kehtib vaid teatud aeg. Järelikult tuleb ajatemplite erinevust uuendada vastava intervalliga.

Rakendusse on implementeeritud intervalliga ajatempli erinevuse uuendamine. Rakendus kasutab kellana funktsiooni *QDateTime::currentMSecsSinceEpoch()*, mis annab süsteemi ajatempli; viimane ei pruugi anda lineaarselt muutuvat aega, kuna süsteemi kell võib muutuda, nt uuendab Interneti kella järgi. On võimalik vältida mitmekordset ajatemplite erinevuse uuesti arvutamist. Selle jaoks on vaja rakenduses kasutada seadme enda kella, mille näit suureneks vaid tegelikult möödunud aja järgi.

6.8 Püsiühendus

Pärast aja sünkroniseerimist üritatakse ühendust üleval hoida. Erinevaid ühendusi on kaks: WiFi ja Bluetooth. Üks ühendus saab aidata teisel ühenduses olla.

Mõlemad pooled hoiavad üksteist kursis WiFi ja Bluetooth sisselülitatusest ning ühendatavuses olemisest. Näiteks kui esimene pool lülitab WiFi välja, siis ei hakka teine pool üritama ühendust taastada, vaid ootab, kuni mõne teise ühenduse (Bluetooth) kaudu teavitatakse, et WiFi on tagasi sisse lülitatud.

Ühendused hoiavad üksteist kursis ka serveri informatsiooniga. Bluetooth juhul pole see vajalik, sest serveri informatsioon ei muutu. WiFi puhul on vaja teavitada, kui serveri informatsioon muutub või puudub, et saaks kiiremini ühenduda, jättes vahele avastamise protsessi. See aitab ühenduse kiiremal taastumisel, kui avastaja ei suuda serverit leida.

Andmeside pärast aja sünkroniseerimist toimub läbi väiksema latentsiga ühenduse. Latents on traadita ühenduse puhul hüppelise väärtusega (ebastabiilne). Seega ei saa usaldada viimast *pingimise* tulemust. Tuleb arvesse võtta ka eelnevad latentsid, et saada üldisem pilt ühenduse latentsist. Hilisemad latentsid on pole nii usaldusväärsed, mis tähendab, et neid tuleb arvestada vähem kui varasemaid latentse. Ühenduse latentside võrdlemisel arvestatakse keskmist latentsi, kus hilisemad on väiksema väärtusega.

Olgu latentside järjend $L = [l_1, l_2, \dots, l_n]$. Ühenduste latentside võrdlemisel kasutatake järjendi viimast elementi l_n. Seade viis läbi *pingimise* ning saab latentsiks l. Sellisel juhul on uus latents l_{n+1} arvutatav valemiga (4).

$$(4)\ l_{n+1} = \frac{l+\sum_{i=1}^{n}\frac{l_i}{w(n-i+1)}}{1+\sum_{i=1}^{n}\frac{1}{w(i)}}$$

Valemis (4) on funktsioon $w(i)$ latentsi kaal, mis on hilisemate latentside korral suurem. Kaal määrab ära kui, jäik ühenduse latents olema peaks. Kui rakenduses on kaal $w(i) = 2^i$ ja järjendi L limiit on 10 elementi – hilisemad visatakse ära.

Tabel 2. Latentsi näide. $w(i) = 2^i$

L enne	Uus tegelik latents	L pärast	Keskmine kaalutud latents
[]	50	[50]	50
[50]	70	[50, 63]	63
[50, 63]	500	[50, 63, 311]	311
[50, 63, 311]	10	[50, 63, 311,100]	100
[50, 63, 311,100]	50	[50, 63, 311,100,97]	97

Kaal $w(i) = 2^i$ ei ole kindlasti absoluutselt õige. Peab arvestama, kui tihti *pingitakse*. Tuleb läbi viia katseid, et valida korrektne kaal. Siin $w(i) = 2^i$ on juhuslikult võetud.

7. Dokumendi sünkroniseerimine

Dokumendi sünkroniseerimisel kasutatakse kõige väiksema latentsiga sünkroniseeritud ühendust.

Sünkroniseerimise alla kuuluvad järgnevad rakenduse osad:

- avatud dokument,
- dokumendis vaadeldav lehekülje number,
- dokumendi peale tehtud märkmed.

Dokumendi valimine ja lehekülje keeramine on ajast sõltuv ning seega tuleb nendele tegevustele lisada aeg. Vastasel korral lähevad seadmed lihtsasti sünkroonist välja. Näiteks seade A keerab leheküljele a, seade B keerab leheküljele b. Kui lehekülje keeramised toimusid ajaliselt lähestikku, siis lõpptulemusena on seadme A ekraanil lehekülg b ja seadme B ekraanil a. Tegelikult peaks mõlemal ekraanil olema kas lehekülg ainult a või ainult b, mis otsustatakse tegevuse aja järgi.

Märkmete tegemist/kustutamist saab teha ajast sõltumatult, lisades igale märkmele identifikaatori ning hoides eraldi enda ja paarilise märkmeid. Eeldus on see, et identifikaator on kasvav ehk igal uuel märkmel on suurim identifikaator. Enda ja paarilise märkmete identifikaatorid võivad kattuda, kuna neid hoitakse eraldi.

Esmasel ühendusel või pärast ühenduse taastamist annavad mõlemad pooled teada, mis dokument neil hetkel on avatud ning mis leheküljel nad on. Kui avatud on sama dokument, siis saadavad mõlemad pooled olemasolevad märkmed, mille järgi otsustatakse millised märkmed alles jätta ja millised kustutada.

Kui paarilisel pole valitud dokumendi olemas, siis see saadetakse talle. Dokumente identifitseeritakse faili sisu SHA1 räsi järgi, et saaks neid võrrelda üle võrgu.

8. Ühenduse analüüs

Analüüsi käigus tehakse kindlaks, kuidas reageerib rakendus ühenduse muutustele, kui stabiilne on ühendus ja milline on sünkroniseeritud aja usaldusväärsus. Jälgitakse mõlema seadme latentsi ja lehekülje pöörde latentsi (aeg, mil lehekülje pööre jõuab kohale teise seadmeni). Lehekülgede latentsid on arvutatud sünkroniseeritud aja järgi, kuna lehekülje keeramise kohalejõudmisest ei teavitata. Aeg sünkroniseeritakse intervalliga 60 sekundit. *Pingimise* intervall on 5 sekundit.

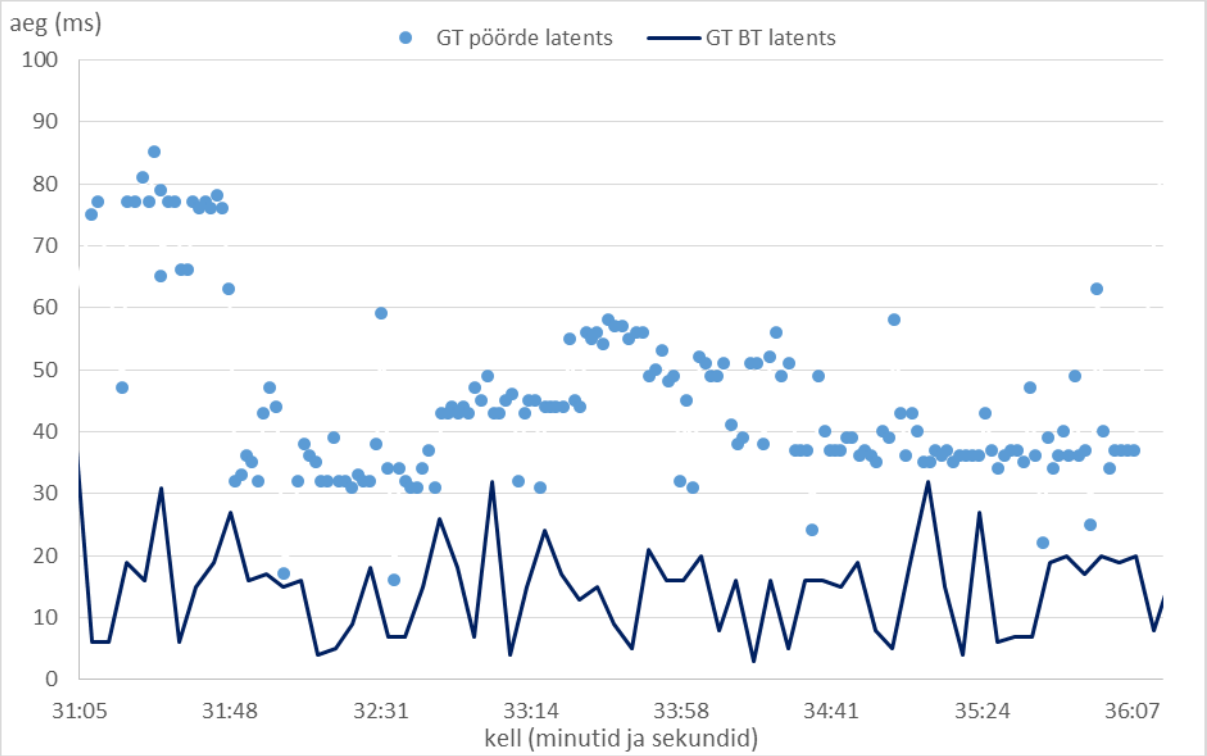
Kõikidel joonistel vertikaalteljeks on tegevuse (*ping*, lehekülje pööre) aeg millisekundites ja horisontaalteljeks on tegevuse läbiviimise kellaaja minutid ja sekundid.

Analüüs viiakse läbi kahe seadme vahel: Samsung I9301I Galaxy S3 (edaspidi GT) ja LG Optimus L3 II E430 (edaspidi LG).

8.1 Pidev Bluetooth

Katse kestis 5 minutit ja 5 sekundit. Seadmed olid sel perioodil üksteise läheduses (<1 m).

GT seade



Joonis 5. Bluetooth GT pöörde latents koos *pingiga*.

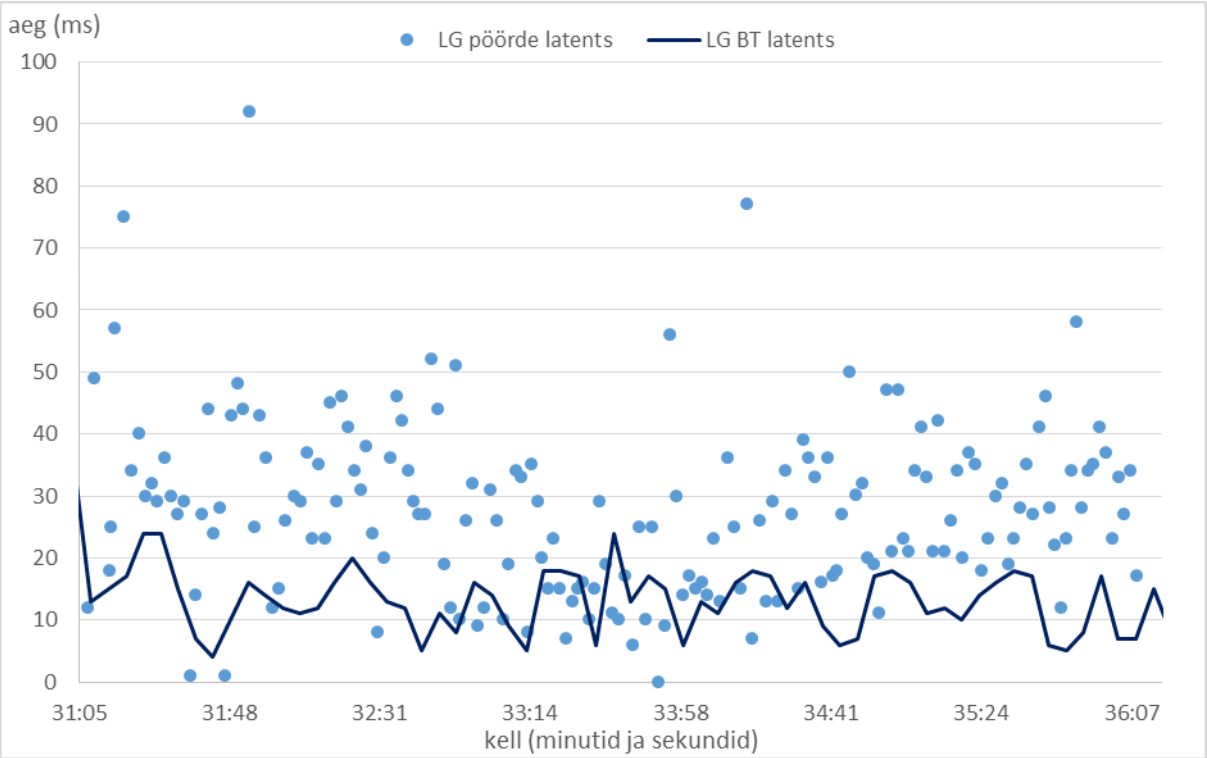
Latentsi maksimum on 32 ms, miinimum 3 ms, keskmine 15 ms ja standardhälve 7 ms. Lehekülje pöörde latentsi maksimum on 85 ms, miinimum 16 ms, keskmine 45 ms ja standardhälve 14 ms.

Nähtav on sünkroniseeritud aja viga, mis tuleb esile arvutatud lehekülje pöörde latentsi väärtuste järgi. On kobarad andmeid. Näiteks vahemik 31:08 – 31:46. Selles osas on lehekülje pöörde latentside keskmine 74 ms. Seadmete aegade erinevus sel hetkel erines keskmisest 31 ms. Järgmine vahemik on 31:50 - 32:47, mille keskmine on 34 ms.

Pöörde latentsid peaksid olema hajutatud latentsi väärtuse ümbruses, sest lehekülje pöörde saatmiseks vajalik andmete suurus on väike ning andmemaht selles rolli ei mängi. Võib

järeldada, et viga on sünkroniseeritud aja tõttu. Sünkroniseeritud aja viga keskmisel juhul on 30 ms.

LG seade



Joonis 6. Bluetooth LG pöörde latents koos *pingiga*.

Latentsi maksimum on 24 ms, miinimum 4 ms, keskmine 13 ms ja standardhälve 5 ms. Lehekülje pöörde latentsi maksimum on 92 ms, miinimum 0 ms, keskmine 28 ms ja standardhälve 14 ms.

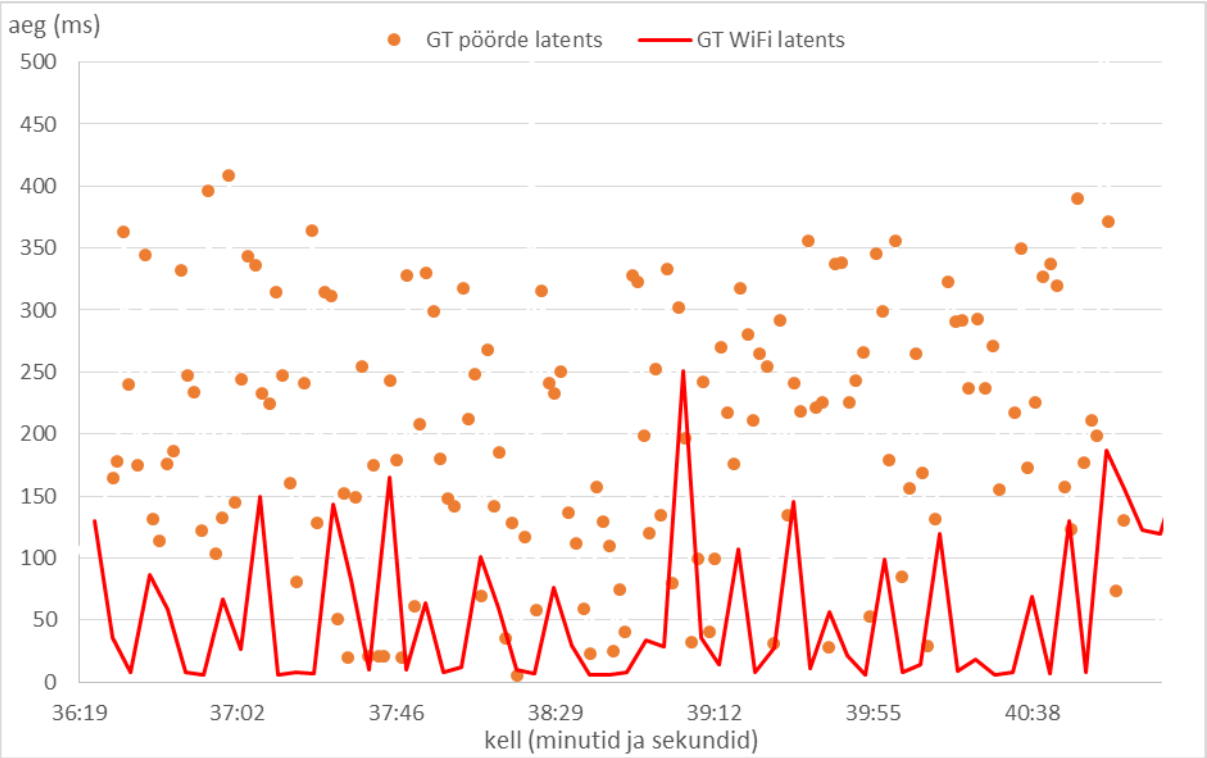
Võib öelda, et Bluetooth latents mõlema seadme poolt *pingides* on samaväärne.

On märkimisväärne, et lehekülje latentsides kobaraid ei ole, aga GT-s oli. Leheküljepöörde latentsi ja latentsi keskmiste erinevus on 15 ms, mis on sünkroniseeritud aja viga keskmisel juhul.

8.2 Pidev WiFi

Katse kestis 4 minutit ja 45 sekundit. WiFi pääsupunkt asus mõlemast seadmest ~10 m kaugusel ning nende vahel oli sein.

GT seade

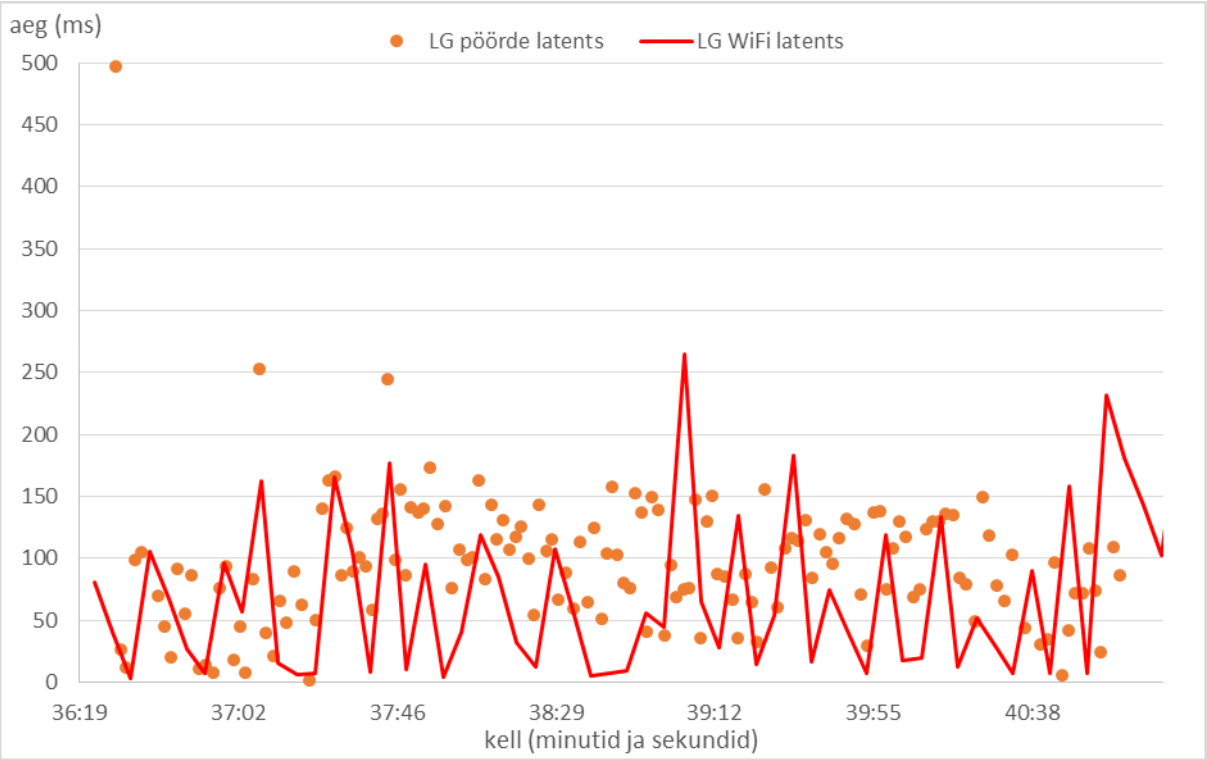


Joonis 7. WiFi GT pöörde latents koos *pingiga*.

Latentsi maksimum on 251 ms, miinimum 6 ms, keskmine 53 ms ja standardhälve 30 ms. Lehekülje pöörde latentsi maksimum on 776 ms, miinimum 5 ms, keskmine 206 ms ja standardhälve 115 ms.

Sünkroniseeritud aja viga keskmisel juhul on 153 ms. See on Bluetooth ühendusest märkimisväärsemalt suurem, mis arvatavasti tuleneb sellest, et WiFi on Bluetoothist ebastabiilsem ja aja sünkroniseerimisel on latentside erinevus suurem.

LG seade



Joonis 8. WiFi LG pöörde latents koos *pingiga*.

Latentsi maksimum on 265 ms, miinimum 3 ms, keskmine 63 ms ja standardhälve 28 ms. Lehekülje pöörde latentsi maksimum on 496 ms, miinimum -25 ms, keskmine 93 ms ja standardhälve 57 ms.

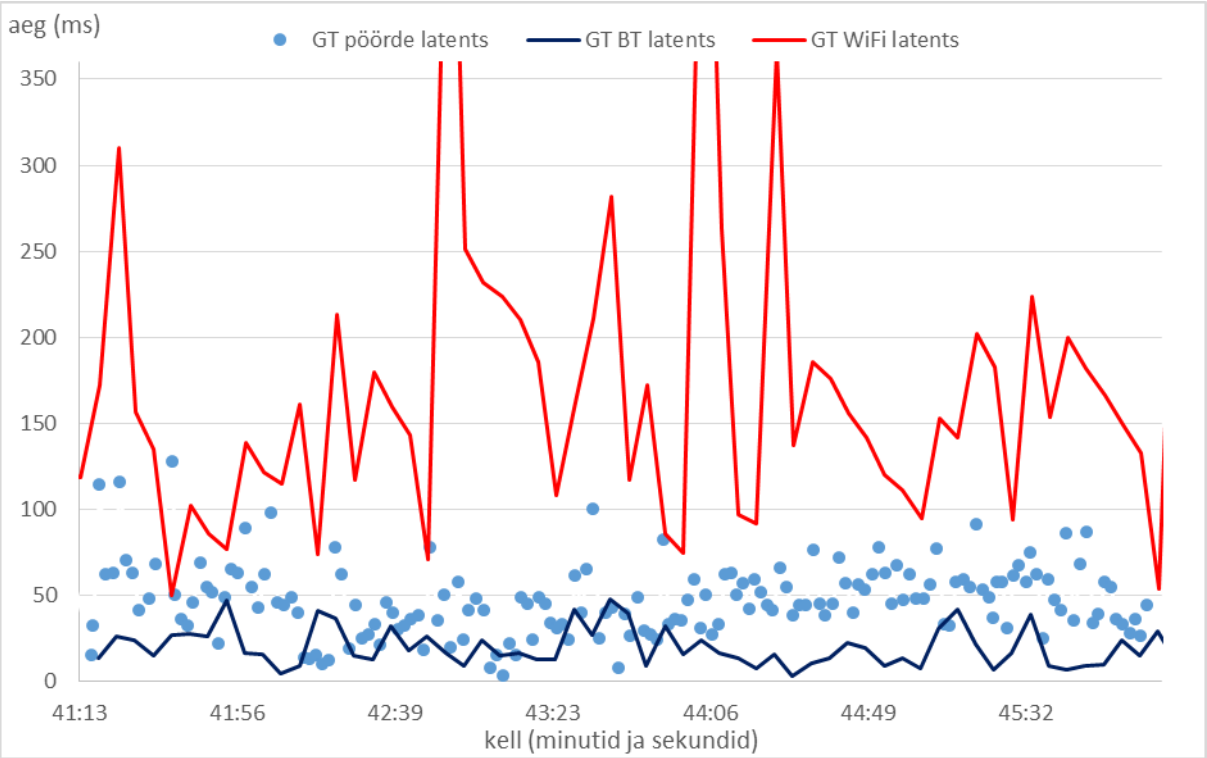
Sünkroniseeritud aja viga keskmisel juhul on 30 ms.

Saime lehekülje pöörde latentsiks -25 ms, mis reaalsuses ei ole negatiivne. Viga tuleneb sünkroniseeritud aja veast.

8.3 Pidev Bluetooth ja WiFi

Katse kestis 4 minutit ja 50 sekundit. Seadmed olid sel perioodil üksteise läheduses (<1 m). WiFi pääsupunkt asus mõlemast seadmest ~10 m kaugusel ning nende vahel oli sein.

GT seade

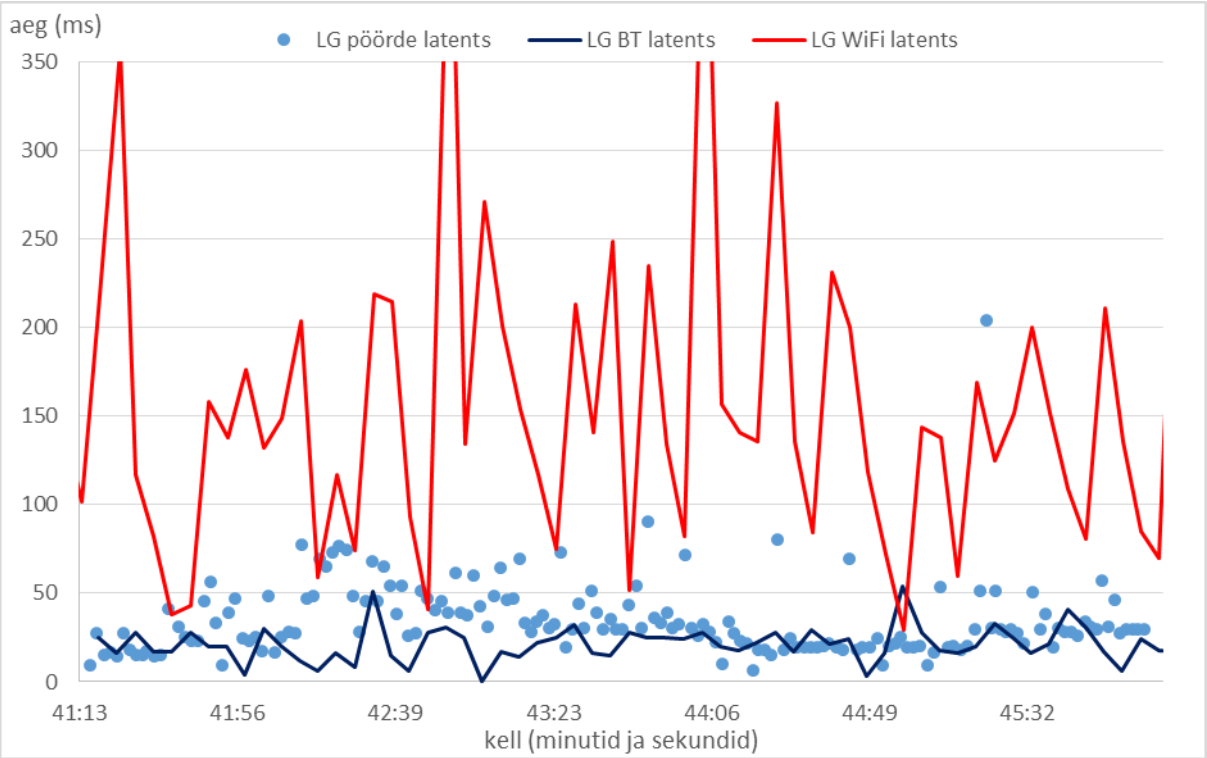


Joonis 9. WiFi ja Bluetooth GT pöörde latents koos *pingiga*.

Bluetooth latentsi maksimum on 48 ms, miinimum 3 ms, keskmine 20 ms ja standardhälve 11 ms. WiFi latentsi maksimum on 578 ms, miinimum 50 ms, keskmine 168 ms ja standardhälve 53 ms. Lehekülje pöörde latentsi maksimum on 128 ms, miinimum 3 ms, keskmine 47 ms ja standardhälve 21 ms.

WiFi ühendus on tunduvalt kõikuvama latentsiga kui Bluetooth ühendus. Standardhälvete erinevus on 42 ms.

LG seade



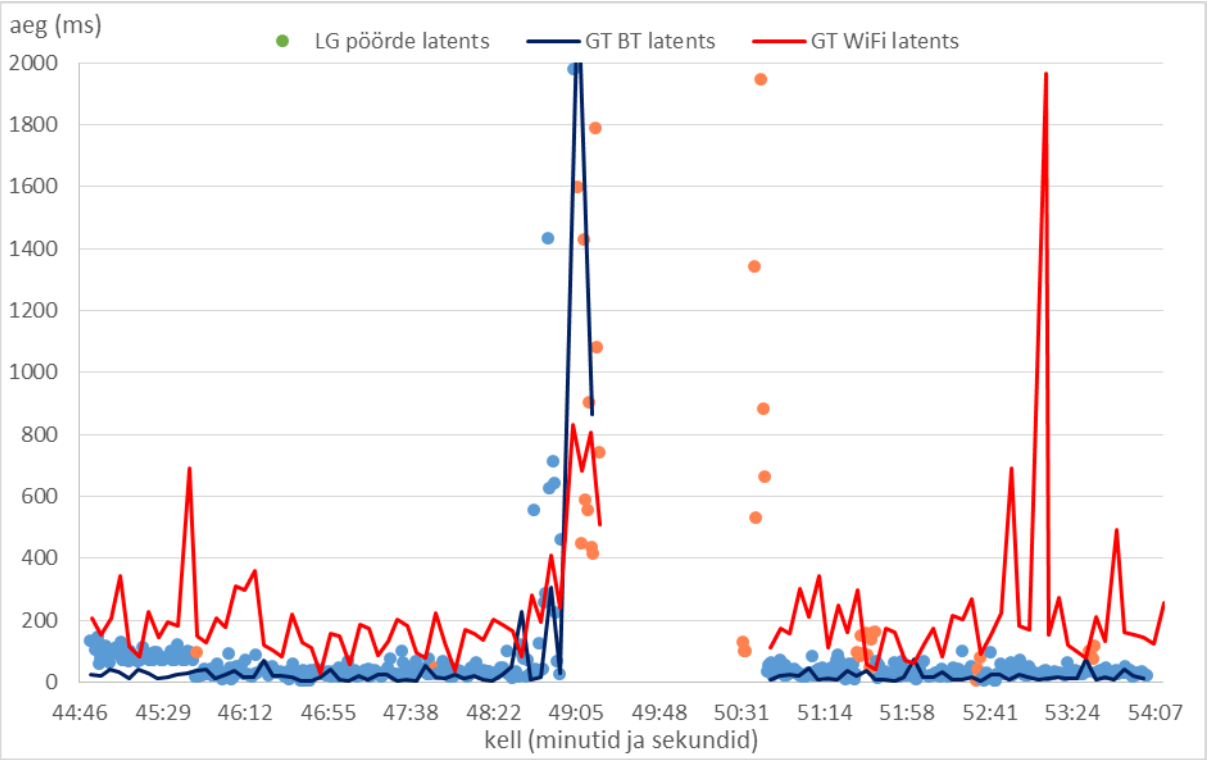
Joonis 10. WiFi ja Bluetooth LG pöörde latents koos *pingiga*.

Bluetooth latentsi maksimum on 54 ms, miinimum 0 ms, keskmine 21 ms ja standardhälve 10 ms. WiFi latentsi maksimum on 488 ms, miinimum 29 ms, keskmine 154 ms ja standardhälve 47 ms. Lehekülje pöörde latentsi maksimum on 204 ms, miinimum 6 ms, keskmine 35 ms ja standardhälve on 21 ms.

8.4 Katkev ühendus

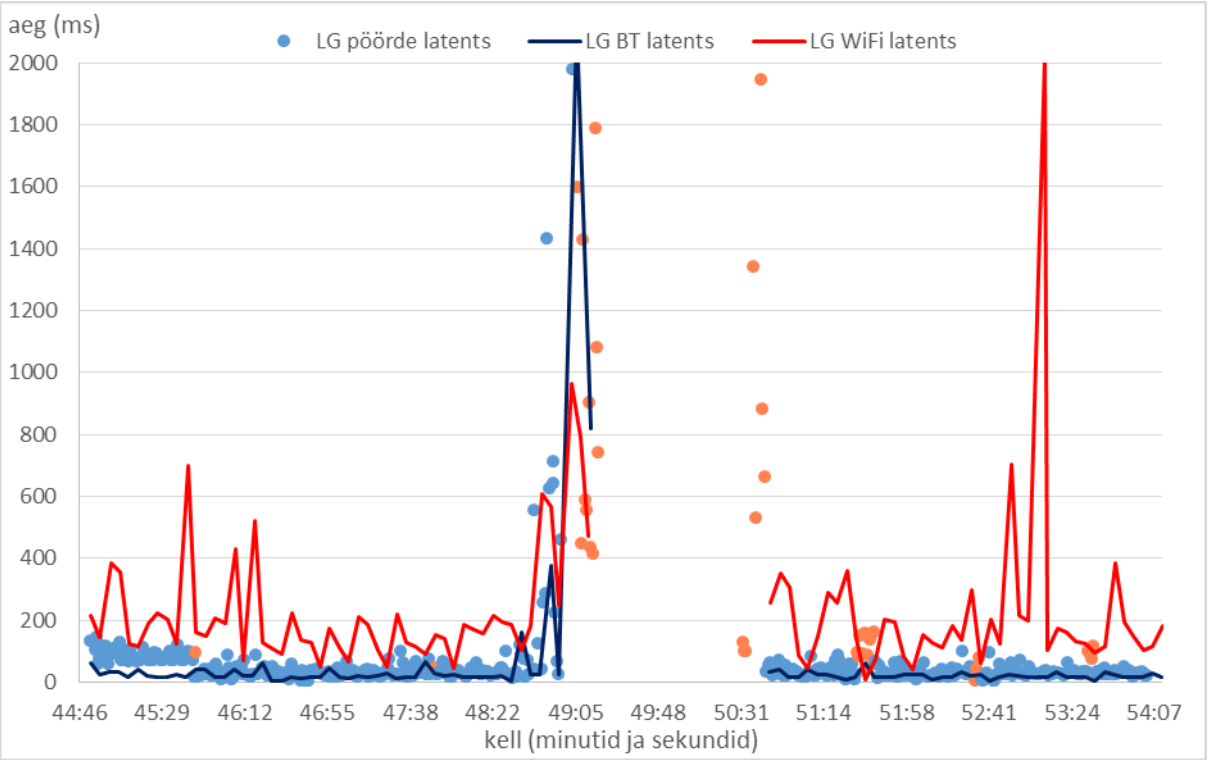
Ühendust ei peatatud/taasalustatud kasutaja poolt. Seadmed taasühendusid iseseisvalt. Ühendus katkes seadmete vahelise kauguse tõttu. Seade GT oli WiFi pääsupunkti juures. Seade LG eemaldus seadmest GT kuni ühenduse katkemiseni. Peale katkemist kaugust vähendati, kuni seade LG oli tagasi seadme GT juures. Lehekülje pöördeid tehti vaid seadmest LG.

GT seade



Joonis 11. GT ühenduse katkemine

LG seade



Joonis 12. LG ühenduse katkemine

Enne ühenduse katkemist tõuseb mõlema seadme latents ja lehekülje pöörde latents märkimisväärselt. Seadmed on võimelised ühendust säilitama üle pikema vahemaa, võrreldes kaugusega, mis on vajalik ühenduse taastamiseks; see järeldub katkemise ja taasühendus hetkede latentside erinevusest. Ühenduse taastumisel on latents ja lehekülje pöörde latents stabiilsed ja madalad. WiFi ja Bluetooth ühendus katkes ja taastus umbes samal ajal.

9. Kvaliteedi tagamine

Testimine viidi läbi vaid Android seadmetel.

9.1 Automaattestid

Automaattestid on olemas osaliselt, mis on saadaval lisas 1. Need testid katavad osa koodist, mis ei vaja ühenduse loomist teise seadmega ja kontakti QML koodiga.

Testimine viidi läbi funktsionaalsuse tasemel rakendades musta kasti meetodit. Kasutati Qt-s olemasolevat testimisteeki.

9.2 Manuaalne testimine

Ühendus

Testid, mis vajavad ühendust teise seadmega, viiakse läbi käsitsi. Selle alla kuulub ühenduse loomine, dokumendi sünkroniseerimine ning ühenduse taastamine selle katkemisel või rakenduse taasavamisel.

Kasutada ei saa tavapäraseid testimisvahendeid, sest tuleb arvestada rohkem kui ühe seadmega. Automaattestide koostamine selle osa jaoks on liiga mahukaks.

QML

QMLiga seotud testimine puudutab kasutajaliidest. Selleks on dokumendi avamine, lehekülgede pööramine ning märkmete loomine ja kustutamine. See osa on testitud käsitsi.

Automaattestide koostamisega tekkis tehnilisi probleeme. QML *unit* testifailidel peab olema eesliides *tst_*. Neid faile testimise alguses üles ei leitud. QML funktsioone saab kutsuda C++ koodis, aga selle läbiviimine tekitas teatud olukordades rakenduse ootamatut peatumist (näiteks dokumendi avamisel).

10. Kokkuvõte

Töö eesmärgiks oli välja arendada kahe nutiseadme vahelist ühendust pidav tarkvaraline rakendus, mis võimaldaks PDF-formaadis failide ühiskasutust. Praktilise rakendusena peeti eeskätt silmas tuge kaheliikmelisele muusikute tiimile nende töös ühise diginoodi alusel.

Loodi töötav rakendus Android seadmetele. Arendusel peeti Androidi kõrval silmas ka iOS platvormi, kuid vajalik iOS spetsiifilise osa käsitlevas töös on üsna lakooniline.

Kõige olulisemal kohal rakenduse juures on seadmete vahelise sünkroniseeritud ühenduse loomine, mis on võimeline katkemise korral taastuma. Selle jaoks üks seade otsib avastamise protsessis teise seadmega ühendumiseks vajaliku informatsiooni. Siis luuakse ühendus teise seadmega, et saaks alustada andmesidet. Seejärel tuleb üksteise turvaline tuvastamine, mis kaitseb vahendusründe eest. Pärast seda vajadusel ühendus krüpteeritakse. Ühenduse viimane etapp on seadmete kellade sünkroniseerimine. Ühenduse loomine töötab vaid lokaalvõrgus.

Teine iseseisev osa rakendusest on dokumentide haldamine, nende sisu kuvamine ekraanil ja selle sünkroniseerimine. Rakenduses olevate dokumentide algformaat on PDF, mis konverteeritakse pildiks, mida saab ekraanil kuvada. Ühenduse olemasolu korral sünkroniseeritakse dokumendi lehekülje numbrit ja märkmeid dokumentide peal. Kui ühendus katkeb, siis taasühendumisel taastatakse sama seis, mis on ühendunud paarilisel.

Tehti põhjalik ühenduse analüüs, kus uuriti andmeside latentsi ja ühenduse muutusi. Ühenduse analüüsi käigus sai kinnituse asjaolu, et rakendus on suuteline katkenud ühendust taastama. Ühendus on heades tingimustes pidev ning ei katke. Tuli ka välja aja sünkroniseerimise keskmine viga.

Eelnevalt polnud autor kokku puutunud kasutatud arendusvahenditega, milleks oli Qt raamistik, programmeerimiseel C++ ja QML. Arusaam seadmete vahelise ühenduse loomisest oli pealiskaudne. Töö käigus saadi palju uusi teadmisi juurde. Eriti raske oli C++ teekide lisamine projekti, kuna need olid kõik lähtekoodina ja arendus toimus Windows operatsioonisüsteemis. Põnev oli Java koodi integreerimine projekti ja OpenSSL teegi kasutamine.

11. Kasutatud materjalid

[1] K. Kruus. (2014) Partiture: An Interoperable Music Stand. [Online]. https://comserv.cs.ut.ee/home/files/MSC2014_KaarelKruus.pdf?study=ATILoputoo&reference=CE523D31A8971262E137BA8148385BB6DF8D1E12

[2] I. Dimitrov. Google Play. [Online]. https://play.google.com/store/apps/details?id=de.im.RemoDroid&hl=en

[3] SAND STUDIO. Google Play. [Online]. https://play.google.com/store/apps/details?id=com.sand.airdroid&hl=en

[4] Tournesol. Google Play. [Online]. https://play.google.com/store/apps/details?id=com.tournesol.tabletremote&hl=en

[5] TeamViewer. Google Play. [Online]. https://play.google.com/store/apps/details?id=com.teamviewer.host.market&hl=en

[6] TeamViewer. Google Play. [Online]. https://play.google.com/store/apps/details?id=com.teamviewer.teamviewer.market.mobile&hl=en

[7] The Qt Company. Qt. [Online]. https://www.qt.io/

[8] Xamarin Inc. Xamarin. [Online]. https://www.xamarin.com/

[9] OpenSSL Software Foundation. OpenSSL. [Online]. https://wiki.openssl.org/index.php/Android

[10] Artifex Software. MuPDF. [Online]. http://mupdf.com/docs/how-to-build-mupdf-for-android

Märkus. Kõikide linkide viimase külastuse aeg on 11.05.2016.

Lisad

I. Projekt

Rakenduse Qt projekt on kaustas *projekt*. Projekt koosneb kahest osast: rakenduse lähtekood, mis on kaustas *src* ja testide lähtekood, mis on kasutas *tests*. Rakenduse lähtekoodi kasutas *src* on olemas eraldi projekt Androidi jaoks, mis on *android_src* kasutas.

II. Kasutusjuhend

Kasutusjuhend on failina *kasutusjuhend.pdf*.

III. Android rakendus

Androidi rakendus on fail *SyncPdf-debug.apk*.

IV. Android installeerimisjuhend

Rakendus Androidile on faili laiendusega *apk*, mis on olemas lisas III.

1. Veendu, et on olemas fail *SyncPdf-debug.apk*.
2. Ühenda Android arvutiga. Kõige lihtsam viis on selleks kasutada USB kaablit.
    a. Ühendus peab olema olekus *media device* (MTP).
3. Lohista *apk* fail arvutist Androidi seadmesse. Selleks sobib Androidi allalaadimiste kaust.
4. Ava kasut, kus *apk* fail paikneb, Androidist failihalduriga.
5. Failihalduris klõpsa *apk* faili peale. Seade küsib nõusolekut *apk* faili installeerimiseks.
6. *apk* fail intalleeritakse Androidile.
    a. Android võib enne installerimist hoiatada, et rakendus on signeeritud tundmatu osapoole poolt ning keeldub rakendust installeerimast.
    b. Hoiatuse aknas saab suunata seadetesse, kus on võimalik lubada installeerida tundmatuid rakendusi.

V. Teadaolevad vead

Kõik vead kerkisid esile testides kahe telefoniga: LG Optimus L3 II E430 ja Samsung I9301I Galaxy S3.

Rakendus

1. Bluetooth ühenduse loomisel peatatakse andmeside WiFi ühenduses. See tähendab, et kui WiFi ühendus on olemas ning hakatakse Bluetooth ühendust looma, siis ei saa läbi WiFi ühenduse andmeid saata. Andmed saadetakse alles pärast Bluetooth ühenduse loomist või ebaõnnestumisel.
2. Kui rakenduse kaudu Bluetooth seade muuta avastatavalt ühendutavaks, siis lülitab Bluetooth seade ennast välja või teeb taaskäivituse.
3. PDFi konverteerimine ja ühenduse andmeside toimub kasutajaliidese lõimest eraldi lõimel. Seega ei tohiks kasutajaliides kokku joosta. Vahepeal kui PDFi konverteeritakse ja samal ajal teisele poolele seda sama faili saadetakse, siis jookseb PDFi konverteeriv seade kokku, kuni teine pool on faili kätte saanud. Kuna seda ei juhtu kogu aeg, siis tõenäoliselt üks ühendustest, WiFi või Bluetooth, blokeerub andmete saatmise ajaks.
4. Suurte failide saatmise ajal võib rakendus krahhida.

5. Ühenduse loomisega võib tekkida probleeme (ei ühendu), kui seade on magavas olekus.
6. WiFi ühendus ei toimi üle Interneti vaid ainult lokaalses võrgus. Selleks, et seadmed saaksid ühenduda üle Interneti, on vaja teada seadmetele antud välist võrgu IP aadressi. Avastamine piirdub ka vaid lokaalse võrguga, kuna võrguseadmed saadavad *broadcast* pakette edasi vaid lokaalses võrgus olevatele seadmetele.

Automaattestid

1. Testiide käivitamisel võib tulla veateade "could not load needed library 'libSyncPdf.so'". Vastav viga tekkis telefoniga LG E430, aga mitte telefoniga Samsung I9301I.

VI. Rakendusega seotud litsentsid

Rakendus kasutab teeke, millel on kaasas litsents. Teek MuPDF litsents on failina *mupdf_litsents.txt*. Teek OpenSSL litsents on failina *openssl_litsents.txt*. Rakenduses olevate nuppude ikooni võeti veebist, millega kaasnes litsents *nuppude_graafika_litsents.pdf*.

VII. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Kevin Nemeržitski**,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose **Sünkroniseeritud dokumentide kuvaja arendus nutiseadmetele**,
(*lõputöö pealkiri*)

mille juhendajad on Kaarel Kruus ja Jüri Kiho,
(*juhendaja nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **12.05.2016**